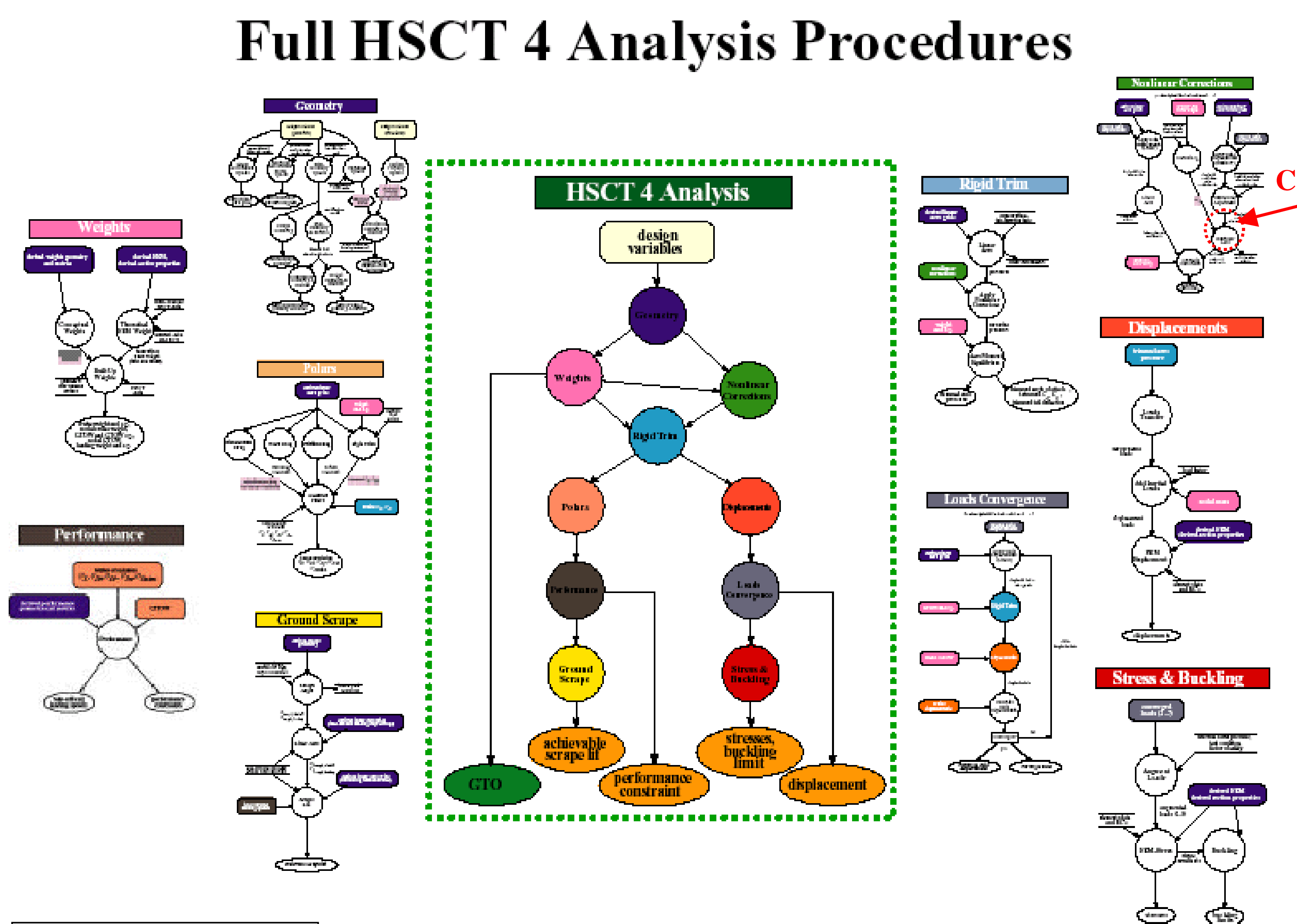
AIAA 2003-3431

Dynamically Reconfigurable Approach to Multidisciplinary Problems

Natalia Alexandrov, NASA LaRC, MDOB
Michael Lewis, College of William & Mary

http://mdob.larc.nasa.gov

Example: a multidisciplinary analysis (MDA)



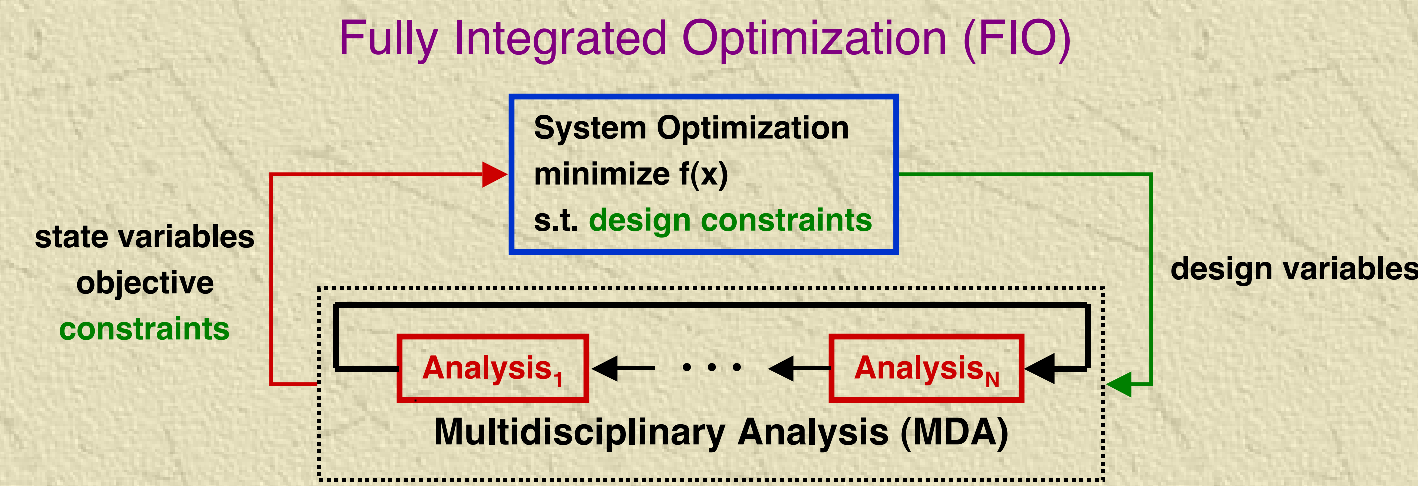
Courtesy J.A. Samareh

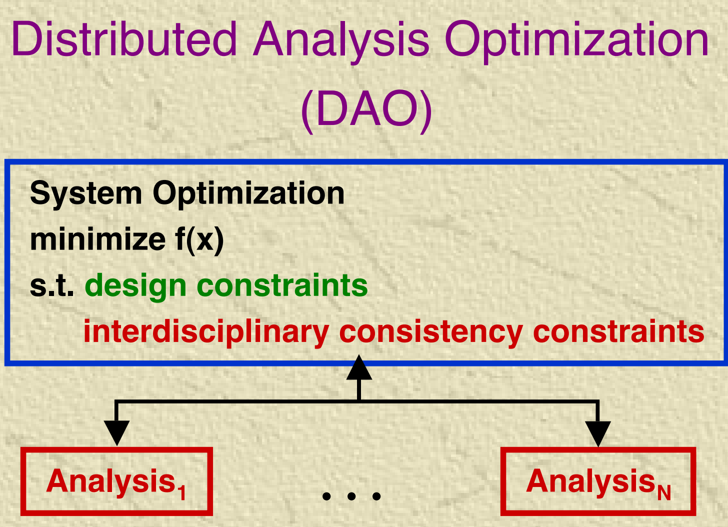
What is reconfigurability?

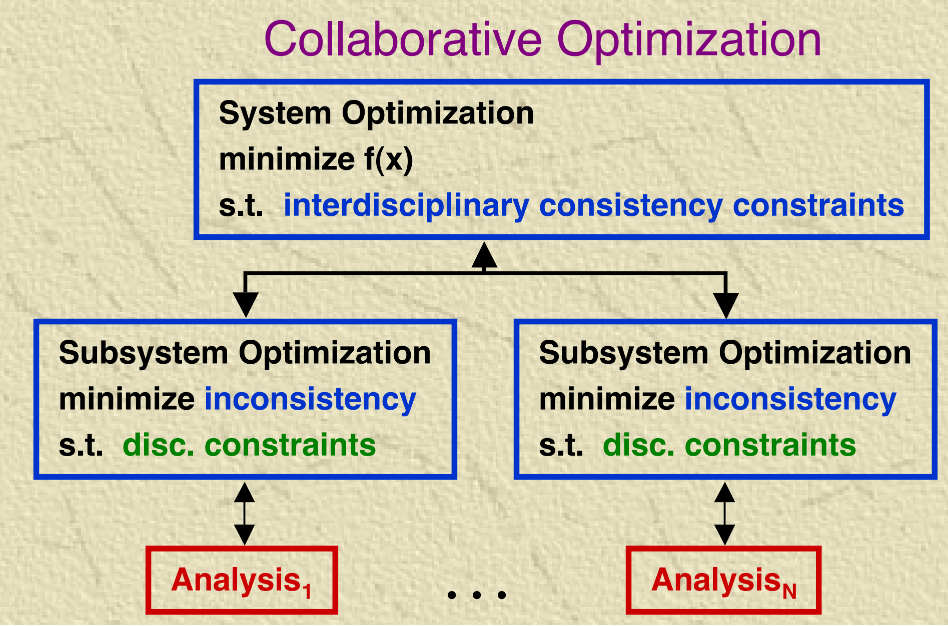
- Computational component-based approach to MDO problem synthesis that allows for straightforward transformation among problem formulations within optimization algorithms
- Assumption: MDO-based NLP $\subsetneq$ design problem
- Outline
  - Effect of problem formulation on tractability
  - Origins of reconfigurability
  - Illustration for 3 formulations and barrier-SQP
  - Long-term plans

Influence of formulation on performance

Example: HPCCP formulation study, Alexandrov & Kodiyalam, AIAA 1998-4884







Influence of formulation on performance, cont.

- Test problems from MDO Test Suite (small, simple)
- Several performance metrics
- Dramatic differences in performance
  - Computational and analytical studies (see paper for refs.): analytical features of formulations, e.g., the degree of disciplinary autonomy, directly affects the ability of numerical algorithms to solve the problem reliably and efficiently

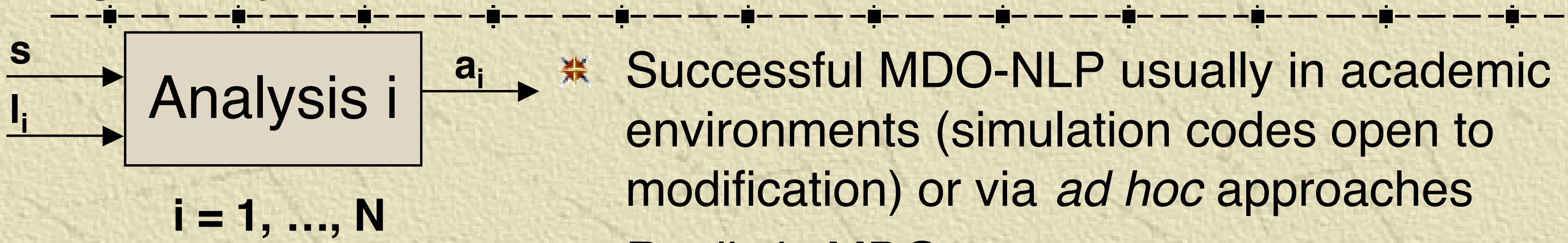
Problem / Method	1	2	3	4	5	6	7
FIO	610	220	610	81	3234	5024	8730
DAO	9530	8976	382	N/A	544	932	N/A
CO	15626	19872	1785	2102	837	40125	691058

Representative # analyses

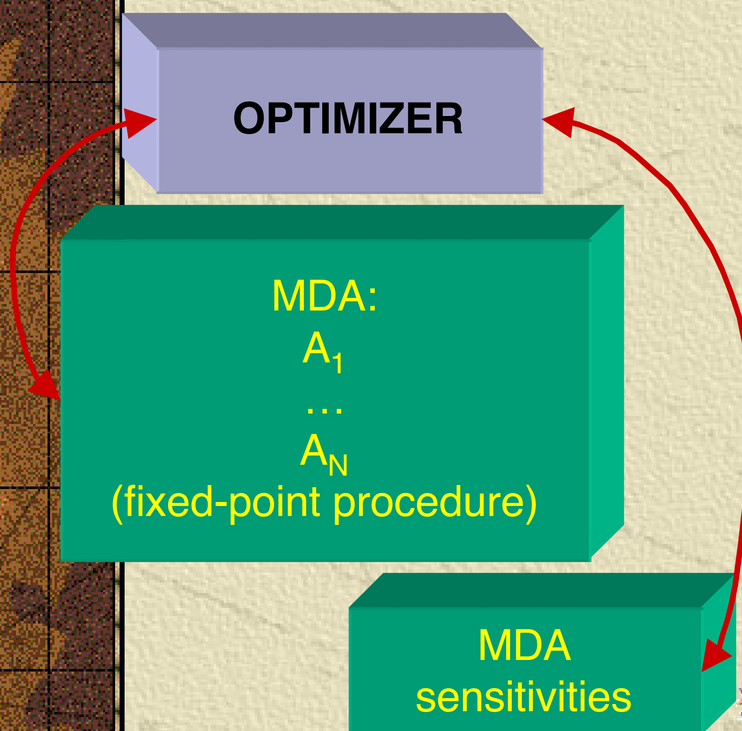
(CO not included here; will consider FIO, DAO, SAND)

MDO Problem Synthesis / Implementation

Problem:
design for objective f with



- Successful MDO-NLP usually in academic environments (simulation codes open to modification) or via *ad hoc* approaches
- Realistic MDO
  - Heroic software integration for MDA
  - MDA = (usually) fixed-point iteration; too rigid
  - May leave no resources for computing derivatives or experimenting with optimization
  - Difficult to get MDA-based objectives and constraints *automatically*
  - To reformulate the problem, need to "unscramble" codes
- ∴One-of-a-kind, monolithic implementations
- Want flexible and/or hybrid re-formulations

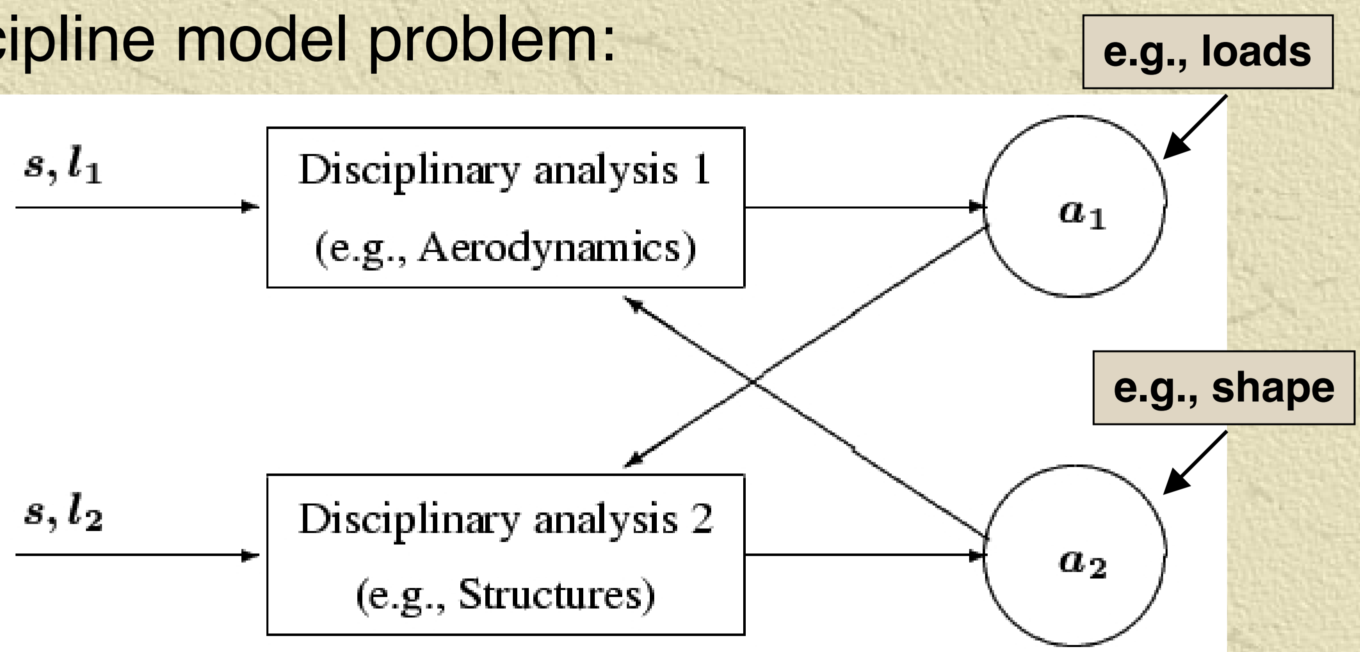


Algorithmic perspective

- Formulation vs. algorithm
- Start with the abilities of available algorithms; devise formulations amenable to algorithms
  - May not satisfy all organizational needs
- Develop reconfigurable approach to synthesis
  - All MDO formulations are related and share the same basic computational components
  - Appropriate implementation enables re-use of components in a straightforward way
- Tools for formulation analysis and matching with algorithms can be included in future computational frameworks

Origins of reconfigurability

- The capacity for reconfigurability stems from the relationship among formulations
- Two-discipline model problem:



- Coupled MDA $\sim$ the physical requirement that a solution satisfy both analyses
- Given $x = (s, l_1, l_2)$, we have

$$\begin{aligned} a_1 &= A_1(s, l_1, a_2) \\ a_2 &= A_2(s, l_2, a_1) \end{aligned}$$

Origins of reconfigurability: SAND

Write MDA as

$$\begin{aligned} a_1 &= A_1(s, l_1, t_2) \\ a_2 &= A_2(s, l_2, t_1) \\ t_1 &= a_1 \\ t_2 &= a_2 \end{aligned}$$

Start with Simultaneous Analysis and Design (SAND) formulation:

(AKA All-at-Once, SAD, etc.)

$$\begin{aligned} &\underset{s, l_1, l_2, a_1, a_2, t_1, t_2}{\text{minimize}} && f(s, t_1, t_2) \\ &\text{subject to} && \underbrace{\left\{ \begin{array}{l} c_1(s, l_1, a_1) \geq 0 \\ c_2(s, l_2, a_2) \geq 0 \end{array} \right.}_{\text{disciplinary constraints}} \\ & && \underbrace{\left\{ \begin{array}{l} a_1 = A_1(s, l_1, t_2) \\ a_2 = A_2(s, l_2, t_1) \end{array} \right.}_{\text{analysis constraints}} \\ & && \underbrace{\left\{ \begin{array}{l} t_1 = a_1 \\ t_2 = a_2 \end{array} \right.}_{\text{consistency constraints}} \end{aligned}$$

Origins of reconfigurability, cont.

- All other formulations may be viewed as derived from the SAND formulation by eliminating a particular set of independent variables from the optimization problem via closing a particular set of constraints or solving optimization problems.

Origins of reconfigurability: DAO

Distributed Analysis Optimization

(AKA Individual Discipline Feasible, In-Between, etc.)

A DAO formulation is

$$
\begin{array}{lll}
\underset{s,l_1,l_2,t_1,t_2}{\text{minimize}} & f(s,t_1,t_2) & \\
\text{subject to} & \left.\begin{array}{l} c_1(s,l_1,t_1) \geq 0 \\ c_2(s,l_2,t_2) \geq 0 \end{array}\right\} & \text{disciplinary constraints} \\
\text{consistency constraints} & \left\{\begin{array}{l} t_1 = a_1(s,l_1,t_2) \\ t_2 = a_2(s,l_2,t_1), \end{array}\right. &
\end{array}
$$

where the disciplinary responses $a_1(s,l_1,t_2)$ and $a_2(s,l_2,t_1)$ are found by closing the disciplinary analysis constraints

$$
\begin{aligned}
a_1 &= A_1(s,l_1,t_2) \\
a_2 &= A_2(s,l_2,t_1).
\end{aligned}
$$

Origins of reconfigurability: FIO

Fully Integrated Optimization (straightforward approach)

The corresponding FIO formulation is

$$\begin{array}{ll} \underset{s,l_1,l_2}{\text{minimize}} & f(s, t_1(s, l_1, l_2), t_2(s, l_1, l_2)) \\ \text{subject to} & c_1(s, l_1, t_1(s, l_1, l_2)) \geq 0 \\ & c_2(s, l_2, t_2(s, l_1, l_2)) \geq 0 \end{array}$$

where we compute $t_1(s, l_1, l_2)$ and $t_2(s, l_1, l_2)$ by solving the MDA

$$\begin{array}{rclcrcl} a_1 & = & A_1(s, l_1, t_2) & \quad & t_1 & = & a_1 \\ a_2 & = & A_2(s, l_2, t_1) & \quad & t_2 & = & a_2. \end{array}$$

Origins of reconfigurability, cont.

- Other formulations further eliminate local design variables by solving disciplinary optimization subproblems
  - Need more work to derive reconfigurable relations
  - Computational components remain unchanged
- Standard results on reduced derivatives will tell us that the sensitivities in DAO and FIO are related to those in SAND *via variable reduction*
- Therefore, computational components of one formulation can be reconfigured to yield those of another

Reduced derivatives

Let

$$\Phi(x) = \phi(x, v(x)).$$

Given x, $v(x)$ is computed from

$$S(x, v(x)) = 0.$$

Let W be the *injection operator* (W^T is the reduction operator):

$$W = W(x, v) = \begin{pmatrix} I \\ -S_v^{-1}(x, v) S_x(x, v) \end{pmatrix}.$$

Define λ by

$$\lambda = \lambda(x, v) = -\left(S_v(x, v)\right)^{-T} \nabla_v \phi(x, v)$$

and the Lagrangian $L(x, v; \lambda)$ by

$$L(x, v; \lambda) = \phi(x, v) + \lambda^T S(x, v).$$

Reduced derivatives

The derivatives of ϕ and Φ are related as follows:

$$\nabla_x \Phi(x) = \underbrace{W^T(x, v(x)) \nabla_{(x,v)} \phi(x, v(x)).}_{\text{Reduced gradient}}$$

$$\nabla^2_{xx} \Phi(x) = \underbrace{W^T \left(\nabla^2_{(x,v)} \phi + \nabla^2_{(x,v)} S \cdot \lambda \right) W,}_{\text{Reduced Hessian of the Lagrangian}}$$

where

$$\begin{aligned} W &= W(x, v(x)) \\ \nabla^2_{(x,v)} \phi &= \nabla^2_{(x,v)} \phi(x, v(x)) \\ \nabla^2_{(x,v)} S \cdot \lambda &= \nabla^2_{(x,v)} S(x, v(x)) \cdot \lambda(x, v(x)) \\ &= \sum_{i=1}^{n} \lambda_i \nabla^2_{(x,v)} S_i . \end{aligned}$$

Barrier-SQP approach to SAND

Now illustrate reconfigurability in the context of a specific class of algorithms: barrier-SQP methods

Let

$$F_{\text{SAND}}(s, l_1, l_2, t_1, t_2) = f(s, t_1, t_2) - \mu \left[\sum_i \ln c_1^i(s, l_1, t_1) + \sum_j \ln c_2^j(s, l_2, t_2) \right]$$

Barrier-SQP solves a sequence of subproblems of the form:

$$\begin{array}{ll} \underset{s, l_1, l_2, t_1, t_2, a_1, a_2}{\text{minimize}} & F_{\text{SAND}}(s, l_1, l_2, t_1, t_2) \\ \text{subject to} & a_1 = A_1(s, l_1, t_2) \\ & a_2 = A_2(s, l_2, t_1) \\ & t_1 = a_1 \\ & t_2 = a_2, \end{array}$$

Barrier-SQP approach to DAO

Let

$$F_{\text{DAO}}(s, l_1, l_2, t_1, t_2) = f(s, t_1, t_2) - \mu \left[\sum_i \ln c_1^i(s, l_1, t_1) + \sum_j \ln c_2^j(s, l_2, t_2) \right]$$

Barrier subproblem for DAO is

$$\begin{array}{ll} \underset{s, l_1, l_2, t_1, t_2}{\text{minimize}} & F_{\text{DAO}}(s, l_1, l_2, t_1, t_2) \\ \text{subject to} & t_1 = a_1(s, l_1, t_2) \\ & t_2 = a_2(s, l_2, t_1), \end{array}$$

where the disciplinary responses $a_1(s, l_1, t_2)$ and $a_2(s, l_2, t_1)$ are computed via the disciplinary analyses:

$$\begin{aligned} a_1 &= A_1(s, l_1, t_2) \\ a_2 &= A_2(s, l_2, t_1). \end{aligned}$$

Relationship among SAND, DAO, FIO Sensitivities

Then setting an appropriate $(\boldsymbol{x}, \boldsymbol{v})$ for each formulation, we have

$$\nabla_{(s,l_1,l_2,t_1,t_2)} \boldsymbol{F}_{\text{DAO}} = \boldsymbol{W}_{\text{DAO}}^{T} \nabla_{(s,l_1,l_2,t_1,t_2,a_1,a_2)} \boldsymbol{F}_{\text{SAND}}$$

and

$$\nabla^2_{(s,l_1,l_2,t_1,t_2)} \boldsymbol{F}_{\text{DAO}} = \boldsymbol{W}_{\text{DAO}}^{T} \nabla^2_{(s,l_1,l_2,t_1,t_2,a_1,a_2)} \boldsymbol{F}_{\text{SAND}} \boldsymbol{W}_{\text{DAO}}.$$

A similar relationship exists between the sensitivities for solving the barrier-SQP subproblems for SAND and FIO:

$$\nabla_{(s,l_1,l_2)} \boldsymbol{F}_{\text{FIO}} = \boldsymbol{W}_{\text{FIO}}^{T} \nabla_{(s,l_1,l_2,t_1,t_2,a_1,a_2)} \boldsymbol{F}_{\text{SAND}}$$

and

$$\nabla^2_{(s,l_1,l_2)} \boldsymbol{F}_{\text{FIO}} = \boldsymbol{W}_{\text{FIO}}^{T} \nabla^2_{(s,l_1,l_2,t_1,t_2,a_1,a_2)} \boldsymbol{F}_{\text{SAND}} \boldsymbol{W}_{\text{FIO}},$$

where the expressions for the reduction operators $\boldsymbol{W}_{\text{FIO}}^{T}$ and $\boldsymbol{W}_{\text{DAO}}^{T}$ are given in the paper.

Solving barrier-SQP subproblem

Solving barrier subproblem is an iterative process, in which we approximately solve

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\boldsymbol{p}^T \boldsymbol{H} \boldsymbol{p} + \boldsymbol{g}^T \boldsymbol{p} \\ \text{subject to} & \nabla \boldsymbol{S}^T \boldsymbol{p} + \boldsymbol{S} = \boldsymbol{0} \end{array}$$

$\boldsymbol{H}$ - approximation to the Hessian of the Lagrangian

$\boldsymbol{g}$ - is the gradient of the Lagrangian

$\boldsymbol{p}$ - step in the iterative process

Reduced-basis approach to barrier-SQP subproblem

- For a specific choice of algorithm for solving the barrier-SQP subproblem, can say even more about the relationship among the computational elements needed to solve the three formulations
- The relationship among the sensitivities means that it is possible to implement an optimization algorithm for SAND so that with a single modification we obtain an algorithm for DAO or FIO

Reduced-basis barrier-SQP for SAND

Algorithm 1: Reduced-basis algorithm for SAND

Initialization: Choose an initial $(\boldsymbol{x}_c, \boldsymbol{v}_c)$.

Until convergence, do {

1. Compute the multiplier $\boldsymbol{\lambda}_{SAND} = -\boldsymbol{S}_v^{-1}\nabla_v \boldsymbol{F}_{\text{SAND}}$.
2. Test for convergence.
3. Construct a local model of $\boldsymbol{L}$ about $(\boldsymbol{x}_c, \boldsymbol{v}_c)$.
4. Take a step $\boldsymbol{p}^{LF}$ to improve linear feasibility:

$$\boldsymbol{p}^{LF} = \alpha \begin{pmatrix} 0 \\ -\boldsymbol{S}_v^{-1}\boldsymbol{S} \end{pmatrix}.$$

5. Subject to the improved linear feasibility, improve optimality:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\boldsymbol{q}^T \boldsymbol{W}^T \boldsymbol{H} \boldsymbol{W} \boldsymbol{q} + (\boldsymbol{g} + \boldsymbol{H}\boldsymbol{p}_{LF})^T \boldsymbol{W}^T \boldsymbol{q} \\ \text{subject to} & \| \boldsymbol{p}_{LF} + \boldsymbol{W}\boldsymbol{q} \| \leq \boldsymbol{r}. \end{array}$$

6. Set $\boldsymbol{p} = (\boldsymbol{p}_x, \boldsymbol{p}_v) = \boldsymbol{p}_{LF} + \boldsymbol{W}\boldsymbol{q}$.
7. Evaluate $(\boldsymbol{x}_+, \boldsymbol{v}_+) = (\boldsymbol{x}_c, \boldsymbol{v}_c) + (\boldsymbol{p}_x, \boldsymbol{p}_v)$ and update $(\boldsymbol{x}_c, \boldsymbol{v}_c)$, $\boldsymbol{r}$. }

Reduced-basis SQP for FIO and DAO

Algorithm 2: Reduced-basis algorithm for SAND + analysis = FIO

Initialization: Choose an initial x_c.

Analysis: Solve $S_{\text{FIO}}(x_c, v_c(x_c)) = 0$ for $v_c(x_c)$.

Until convergence, do {

1–6. These steps remain unchanged.

7. **Analysis: Solve $S_{\text{FIO}}(x_+, v_+) = 0$ for $v_+(x_+)$; evaluate (x_+, v_+).**

8. This step remains unchanged.

}

Algorithm 3: Reduced-basis algorithm for SAND + analysis = DAO

Initialization: Choose an initial (x_c, v_c).

Analysis: Solve $S_{\text{DAO}}(x_c, v_c(x_c)) = 0$ for $v_c(x_c)$.

Until convergence, do {

1–6. These steps remain unchanged.

7. **Analysis: Solve $S_{\text{DAO}}(x_+, v_+) = 0$ for $v_+(x_+)$; evaluate (x_+, v_+).**

8. This step remains unchanged.

}

Other algorithms

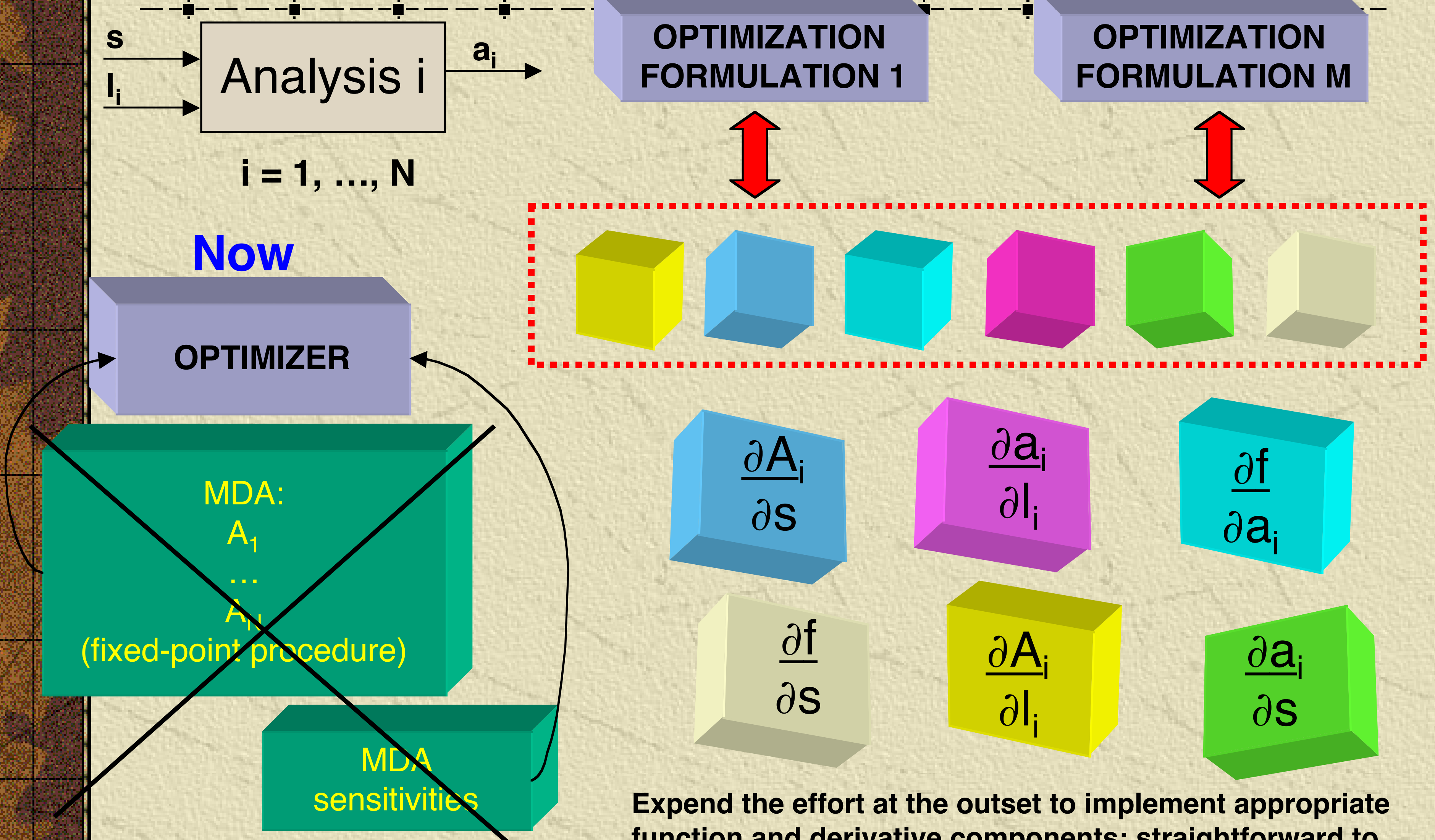
- Outlined reconfigurable scheme should work for other methods that handle inequalities via a penalty function (e.g., augmented Lagrangian)
- Active set methods are likely to take more work

Concluding remarks

- MDO problem formulation directly affects the tractability of the problem
- There are many formulations with a spectrum of benefits
- Regardless of the formulation or even the paradigm, there is a clear need for flexible problem synthesis and easy reconfiguration
- Basic computational components combined with transformations within specific algorithms form a promising approach
- Plan: develop tools for analysis of problems in terms of formulation and algorithm matching

MDO Problem Synthesis / Implementation

Problem:
design for objective f with



Laborious, expensive, one-time integration, difficult to transform/ expand

Expend the effort at the outset to implement appropriate function and derivative components; straightforward to transform and expand: an opportunity for a general framework